IASD M2 at Paris Dauphine

Deep Reinforcement Learning

12: Model-Based Reinforcement Learning

Eric Benhamou - Thérèse des Escotais

PSL UNIVERSITÉ PARIS Dauphine UNIVERSITÉ PARIS ENS ÉCOLE NORMALE SUPÉRIEURE MINES ParisTech

Homework 3 : Q-Learning and Actor-Critic Algorithms

Due on Wed 28 February.

3 outputs to submit:

1. Report (pdf)
2. (code) Submit.zip
3. Google colab notebook

Any homework submitted late will not be graded

Ask your questions on Moodle and answer to others

Acknowledgement

These materials are based on the seminal course of Sergey Levine CS285

Today's Lecture

1. Basics of model-based RL: learn a model, use model for control
   - Why does naïve approach not work?
   - The effect of distributional shift in model-based RL
2. Uncertainty in model-based RL
3. Model-based RL with complex observations
4. Next time: **policy learning** with model-based RL

- Goals:
  - Understand how to build model-based RL algorithms
  - Understand the important considerations for model-based RL
  - Understand the tradeoffs between different model class choices

Why learn the model?

If we knew $f(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{s}_{t+1}$, we could use the tools from last week.

(or $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ in the stochastic case)

So let's learn $f(\mathbf{s}_t, \mathbf{a}_t)$ from data, and *then* plan through it!

model-based reinforcement learning version 0.5:

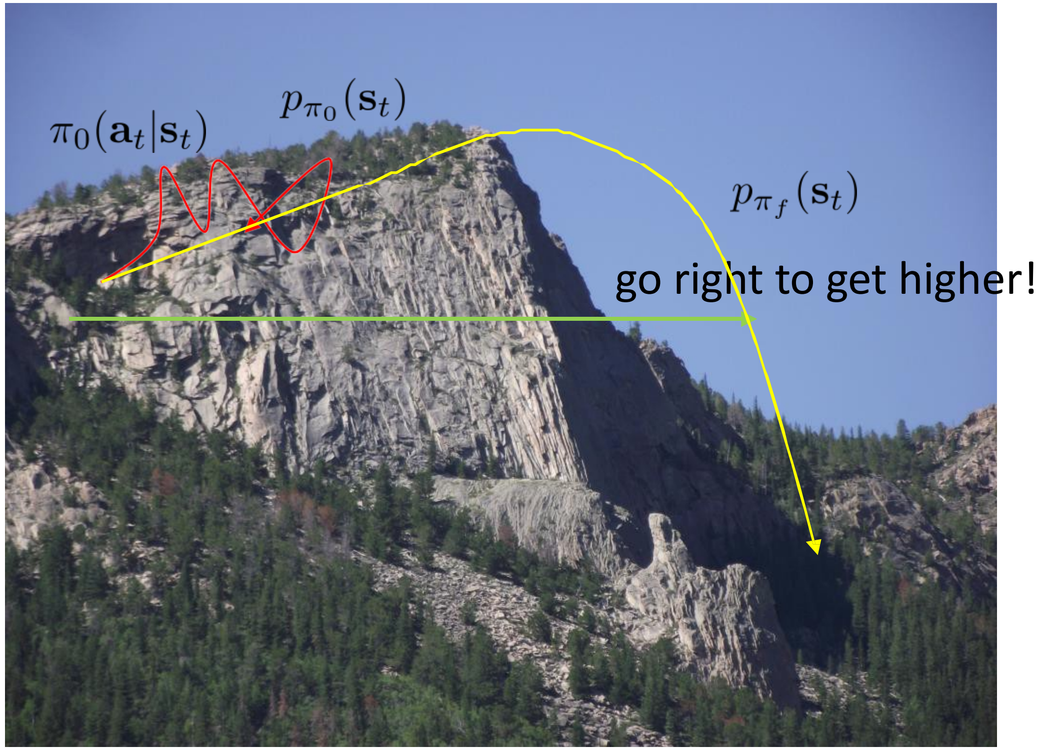
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions

Does it work? Yes!

- Essentially how system identification works in classical robotics
- Some care should be taken to design a good base policy
- Particularly effective if we can hand-engineer a dynamics representation using our knowledge of physics, and fit just a few parameters

Does it work?

No!



1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions

$$p_{\pi_f}(\mathbf{s}_t) \neq p_{\pi_0}(\mathbf{s}_t)$$

- Distribution mismatch problem becomes exacerbated as we use more expressive model classes

Can we do better?

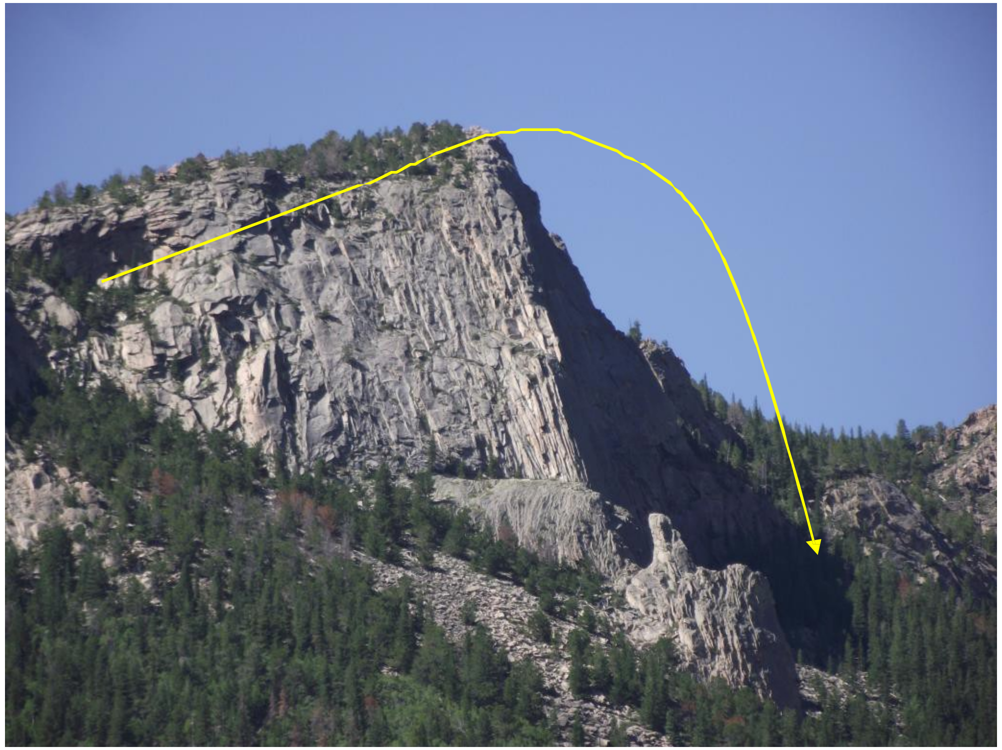
can we make $p_{\pi_0}(\mathbf{s}_t) = p_{\pi_f}(\mathbf{s}_t)$?

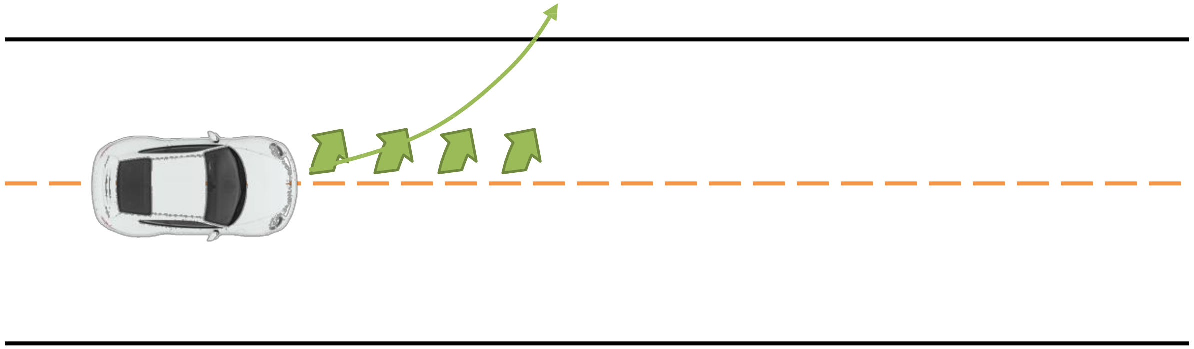
where have we seen that before? need to collect data from $p_{\pi_f}(\mathbf{s}_t)$

model-based reinforcement learning version 1.0:

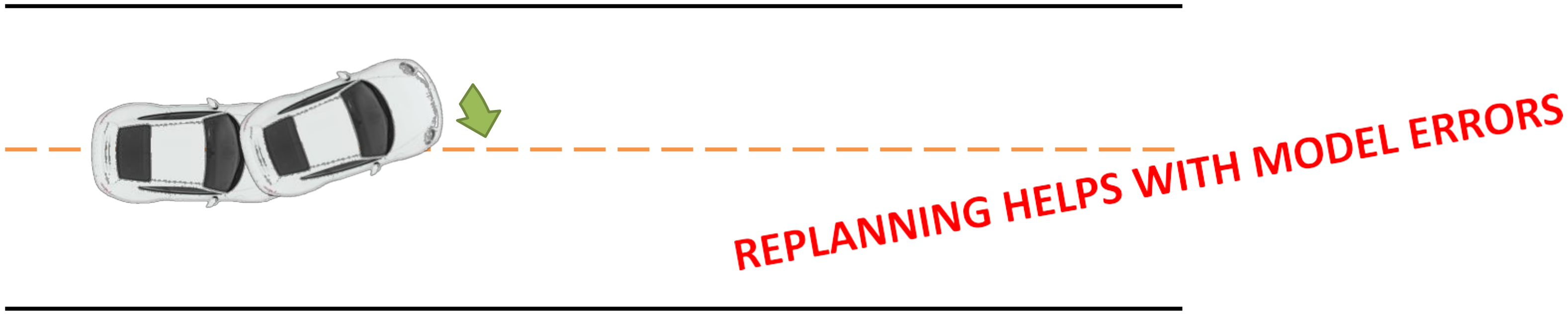
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute those actions and add the resulting data $\{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_j\}$ to $\mathcal{D}$

What if we make a mistake?





Can we do better?



model-based reinforcement learning version 1.5:

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$

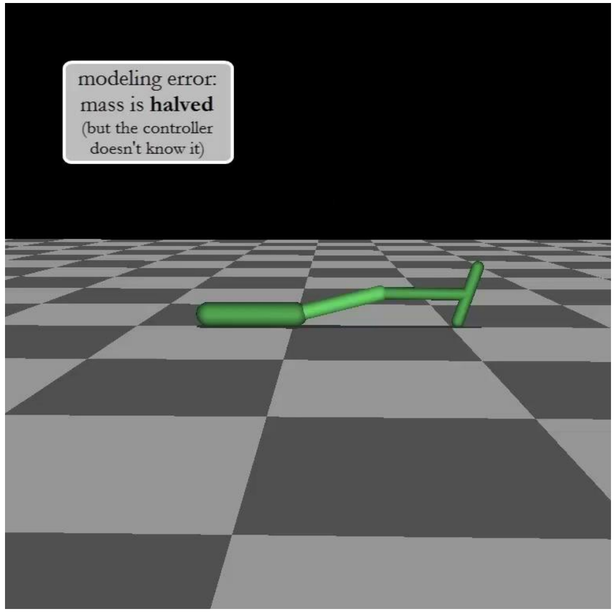
How to replan?

model-based reinforcement learning version 1.5:

every N steps

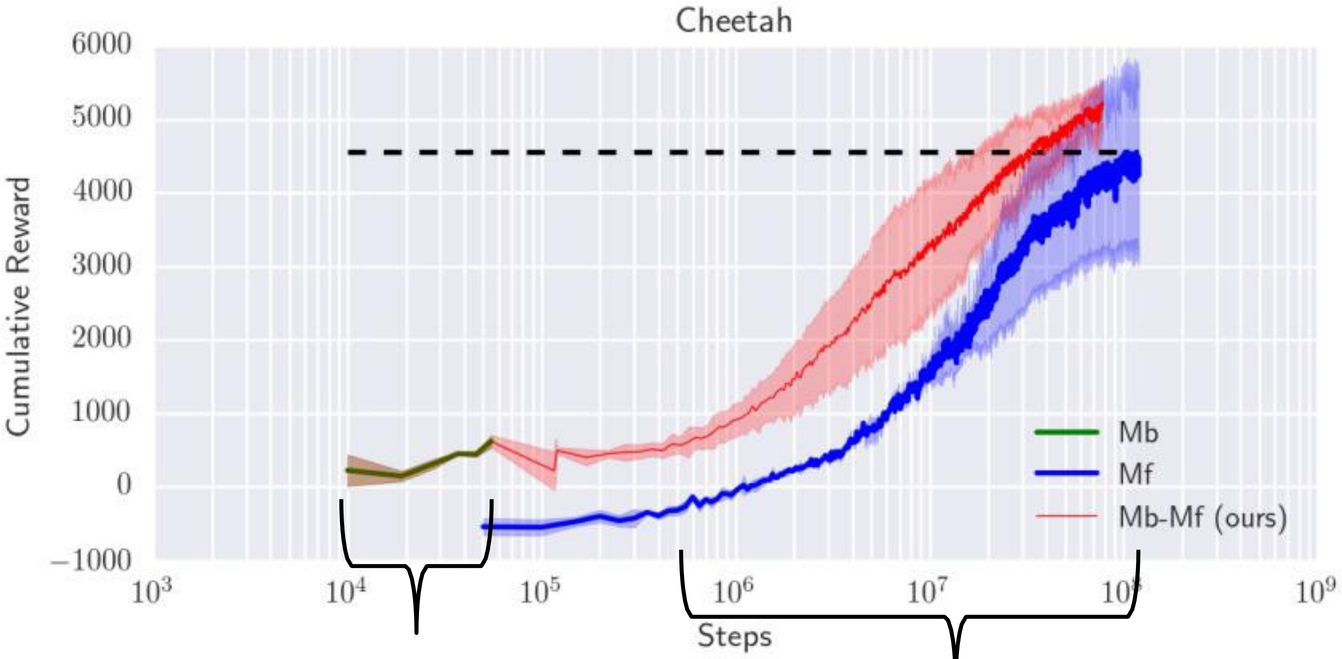
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$

- The more you replan, the less perfect each individual plan needs to be
- Can use shorter horizons
- Even random sampling can often work well here!



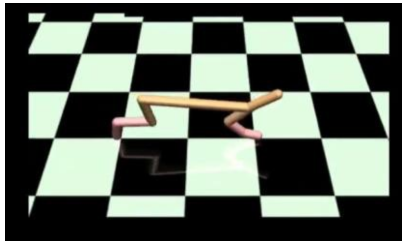
Uncertainty in Model-Based RL

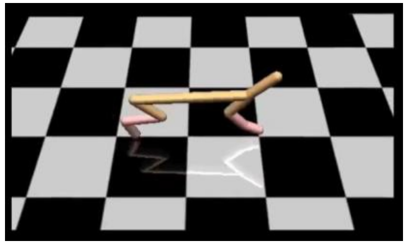
A performance gap in model-based RL



pure model-based
(about 10 minutes real time)

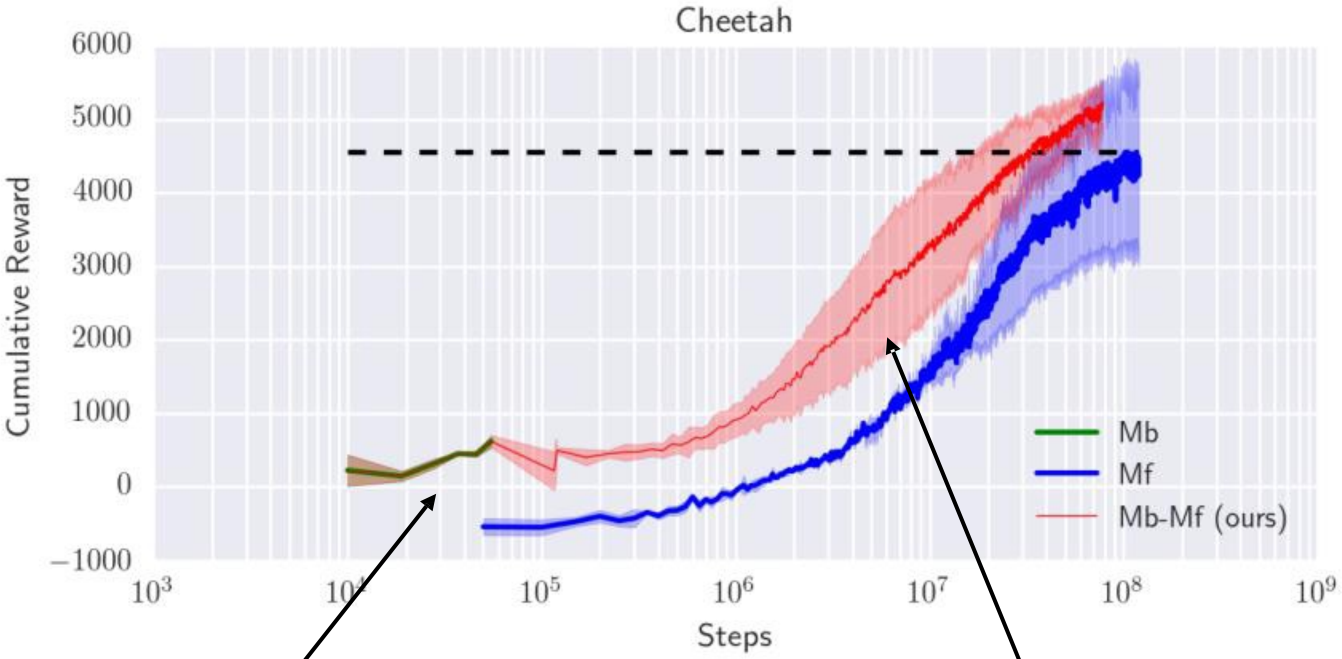
model-free training
(about 10 days…)





Nagabandi, Kahn, Fearing, L. ICRA 2018

Why the performance gap?



need to not overfit here…

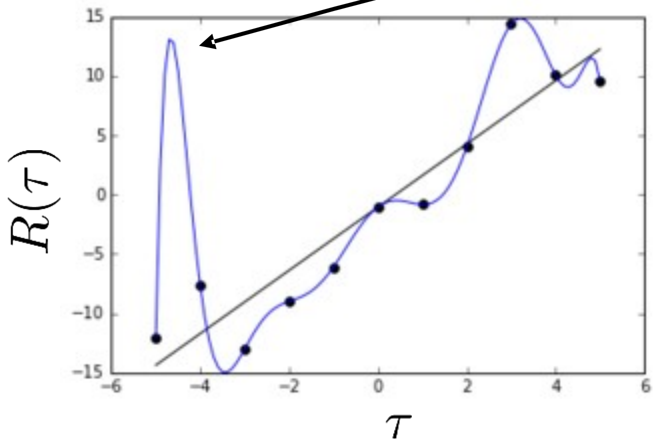
…but still have high capacity over here

Why the performance gap?

model-based reinforcement learning version 1.5:

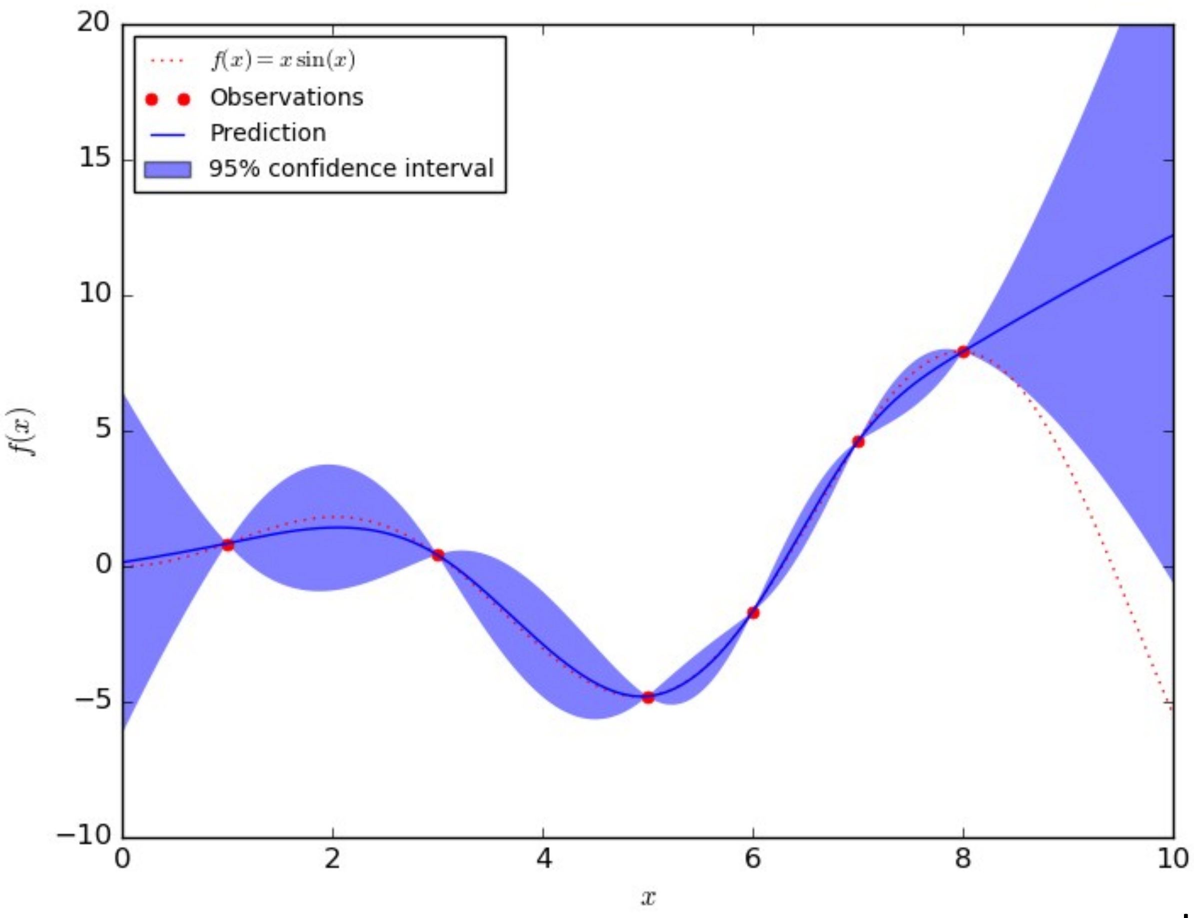
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$

every N steps

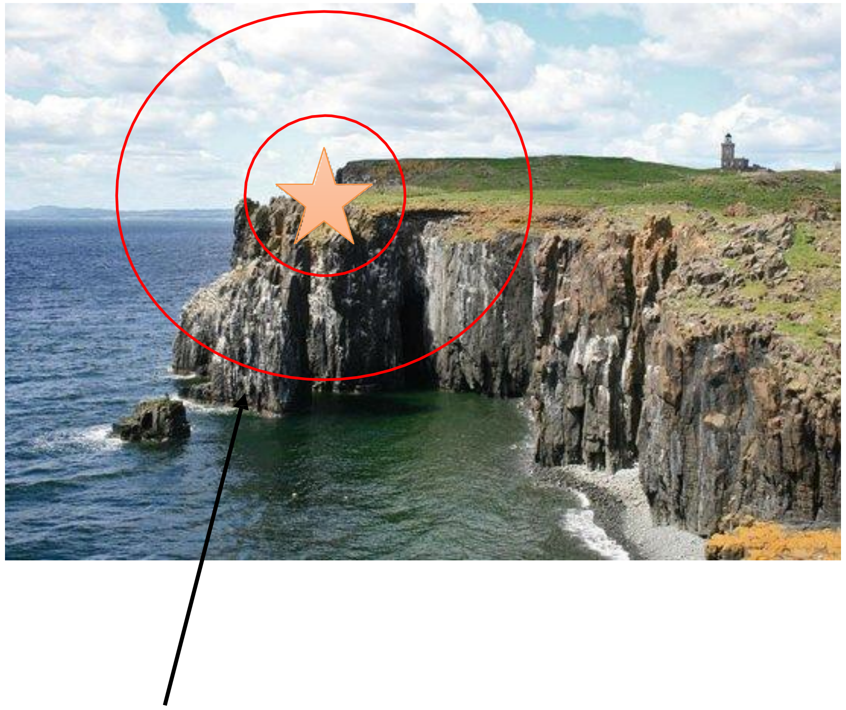


very tempting to go here…

How can uncertainty estimation help?



$$p_{\pi_f}(\mathbf{s}_t) \neq p_{\pi_0}(\mathbf{s}_t)$$



expected reward under high-variance prediction is **very** low, even though mean is the same!

Intuition behind uncertainty-aware RL

model-based reinforcement learning version 1.5:

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$

every N steps

only take actions for which we think we'll get high reward in expectation (w.r.t. uncertain dynamics)

This avoids "exploiting" the model

The model will then adapt and get better

There are a few caveats…

Need to explore to get better

Expected value is not the same as pessimistic value

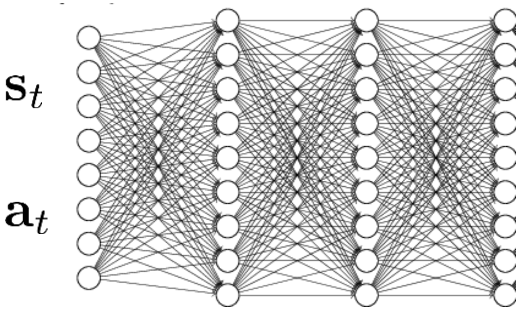
Expected value is not the same as optimistic value

…but expected value is often a good start

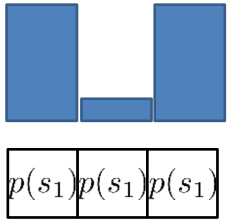
Uncertainty-Aware Neural Net Models

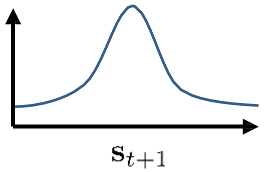
How can we have uncertainty-aware models?

Idea 1: use output entropy

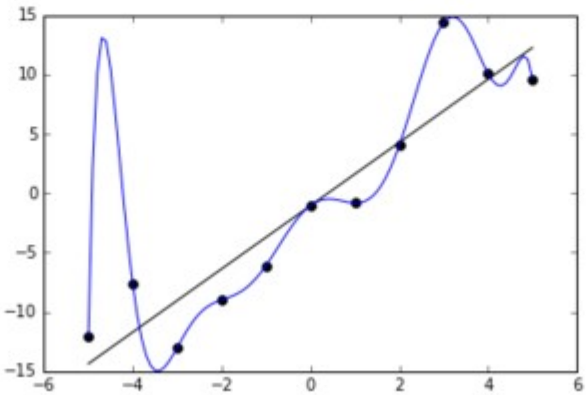


$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$



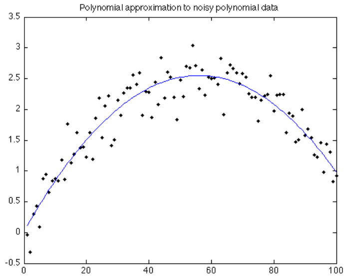


why is this not enough?



Two types of uncertainty:

aleatoric or *statistical* uncertainty



epistemic or *model* uncertainty

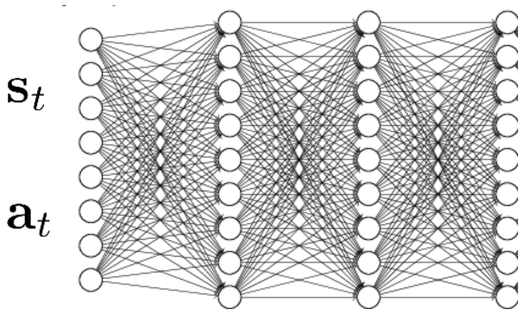
"the model is certain about the data, but we are not certain about the model"

what is the variance here?

How can we have uncertainty-aware models?

Idea 2: estimate model uncertainty

"the model is certain about the data, but we are not certain about the model"



$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$

parameters θ

usually, we estimate

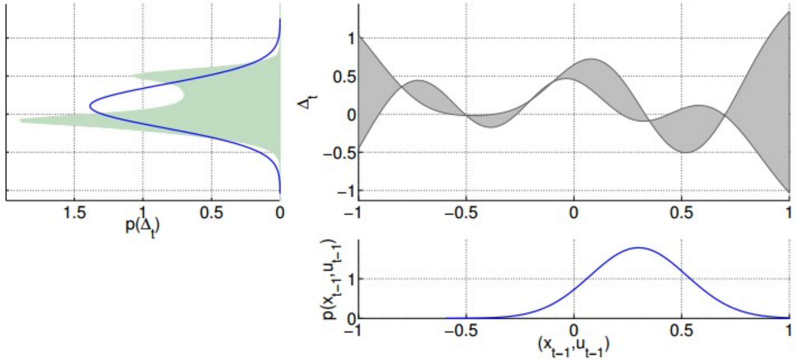
$$\arg\max_\theta \log p(\theta|\mathcal{D}) = \arg\max_\theta \log p(\mathcal{D}|\theta)$$

can we instead estimate $p(\theta|\mathcal{D})$?

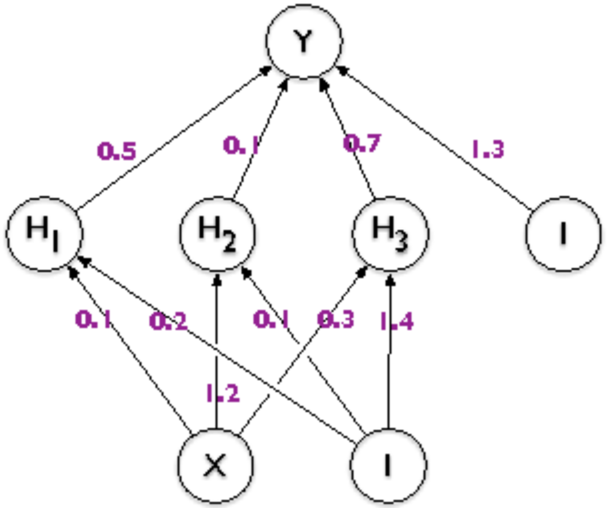
the entropy of this tells us the model uncertainty!

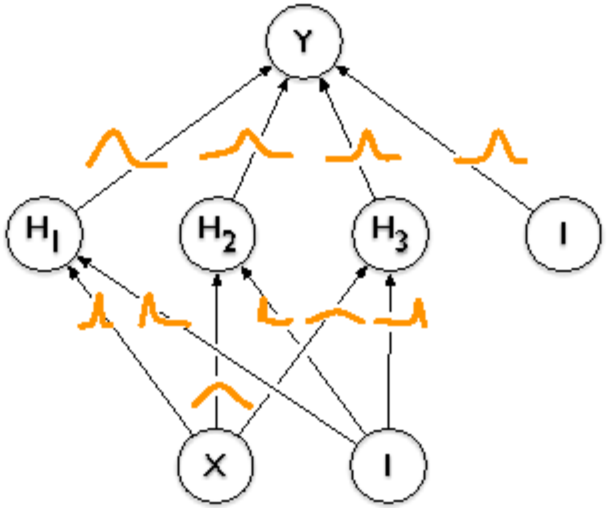
predict according to:

$$\int p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta) p(\theta|\mathcal{D}) d\theta$$



Quick overview of Bayesian neural networks





common approximation:

$$p(\theta|\mathcal{D}) = \prod_i p(\theta_i|\mathcal{D})$$

$$p(\theta_i|\mathcal{D}) = \mathcal{N}(\mu_i, \sigma_i)$$

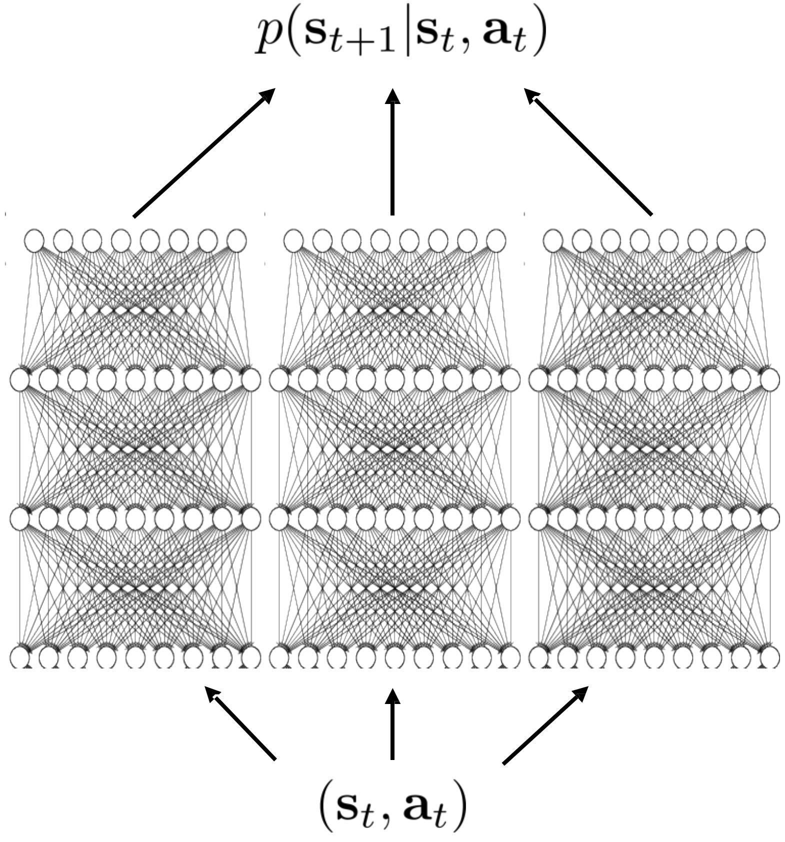
expected weight

uncertainty
about the weight

For more, see:
Blundell et al., Weight Uncertainty in Neural Networks
Gal et al., Concrete Dropout

We'll learn more about variational inference later!

Bootstrap ensembles



Train multiple models and see if they agree!

$$\text{formally:} \quad p(\theta|\mathcal{D}) \approx \frac{1}{N} \sum_i \delta(\theta_i)$$

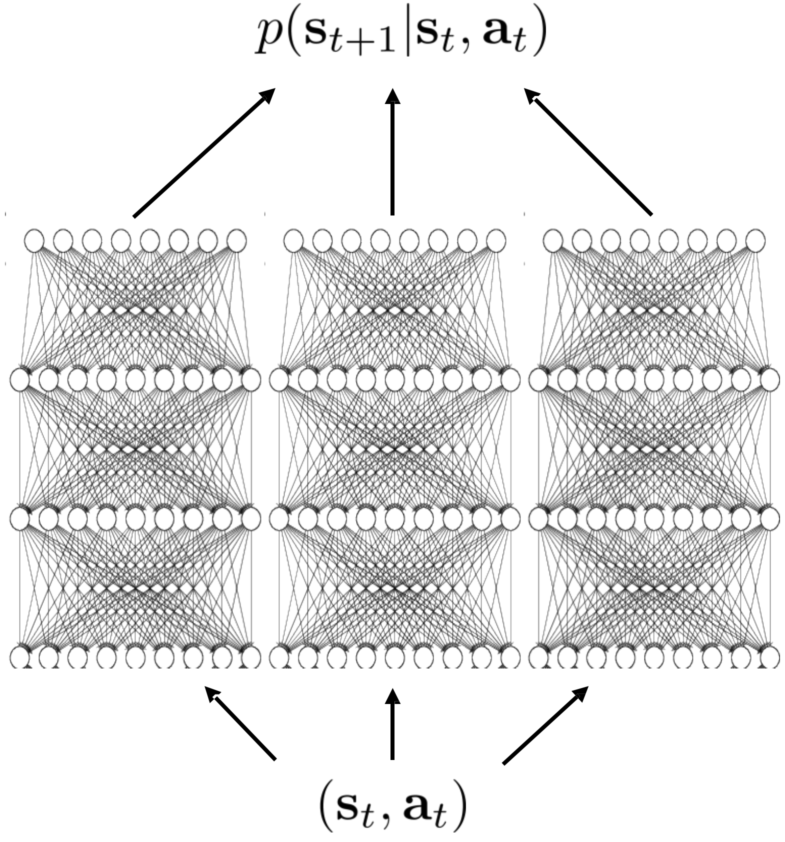
$$\int p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta) p(\theta|\mathcal{D}) d\theta \approx \frac{1}{N} \sum_i p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta_i)$$

How to train?

Main idea: need to generate "independent" datasets to get "independent" models

θ_i is trained on $\mathcal{D}_i$, sampled *with replacement* from $\mathcal{D}$

Bootstrap ensembles in deep learning



This basically works

Very crude approximation, because the number of models is usually small (< 10)

Resampling with replacement is usually unnecessary, because SGD and random initialization usually makes the models sufficiently independent

Planning with Uncertainty, Examples

How to plan with uncertainty

Before: $J(\mathbf{a}_1, \dots, \mathbf{a}_H) = \sum_{t=1}^{H} r(\mathbf{s}_t, \mathbf{a}_t)$, where $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$

Now: $J(\mathbf{a}_1, \dots, \mathbf{a}_H) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{H} r(\mathbf{s}_{t,i}, \mathbf{a}_t)$, where $\mathbf{s}_{t+1,i} = f_i(\mathbf{s}_{t,i}, \mathbf{a}_t)$

distribution over deterministic models

In general, for candidate action sequence $\mathbf{a}_1, \dots, \mathbf{a}_H$:

Step 1: sample $\theta \sim p(\theta|\mathcal{D})$

Step 2: at each time step t, sample $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta)$

Step 3: calculate $R = \sum_t r(\mathbf{s}_t, \mathbf{a}_t)$

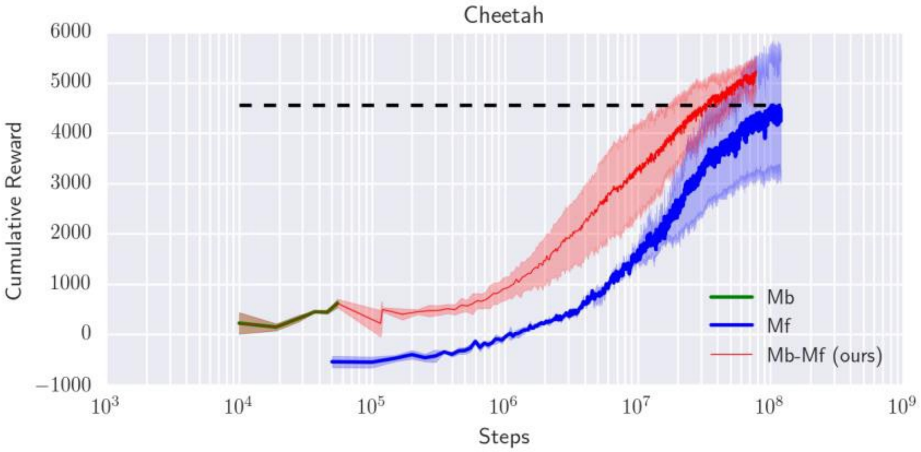
Step 4: repeat steps 1 to 3 and accumulate the average reward

Other options: moment matching, more complex posterior estimation with BNNs, etc.

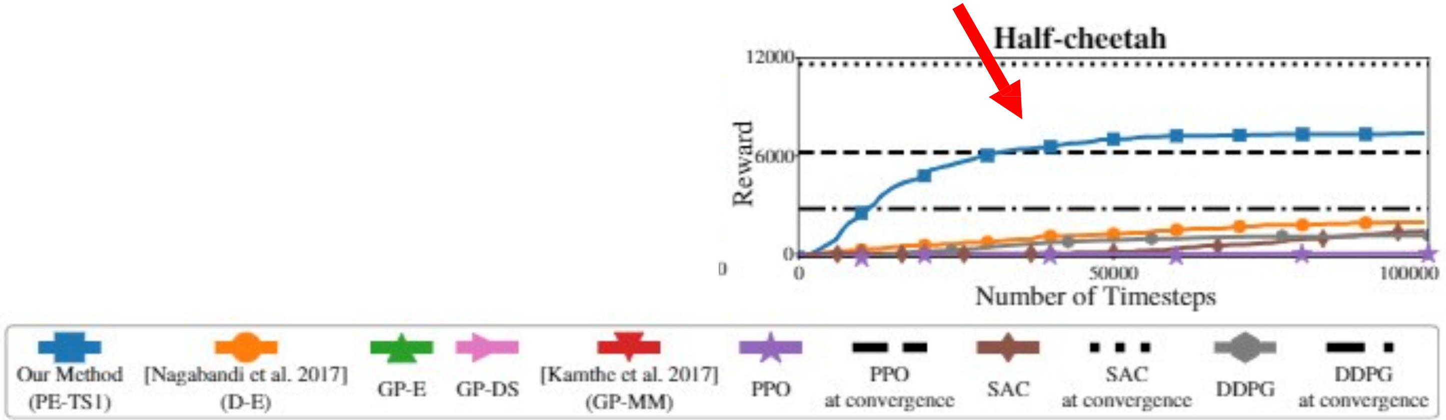
Example: model-based RL with ensembles

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models

exceeds performance of model-free after 40k steps (about 10 minutes of real time)



before



after

More recent example: PDDM

Deep Dynamics Models for Learning Dexterous Manipulation. Nagabandi et al. 2019

Further readings

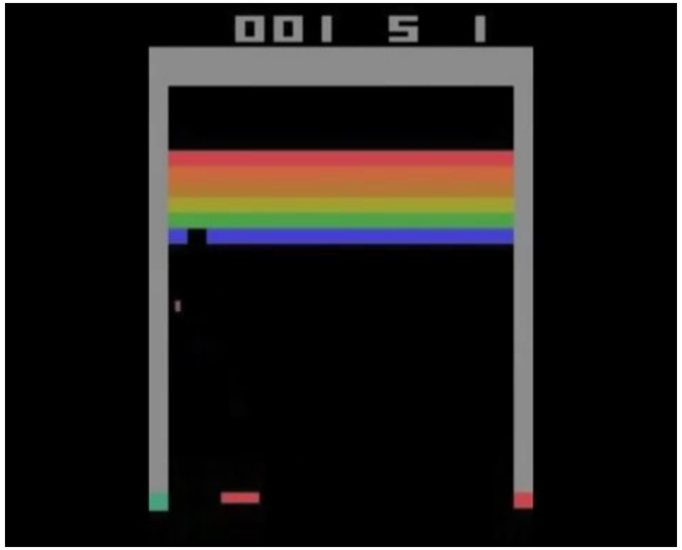
- Deisenroth et al. PILCO: A Model-Based and Data-Efficient Approach to Policy Search.

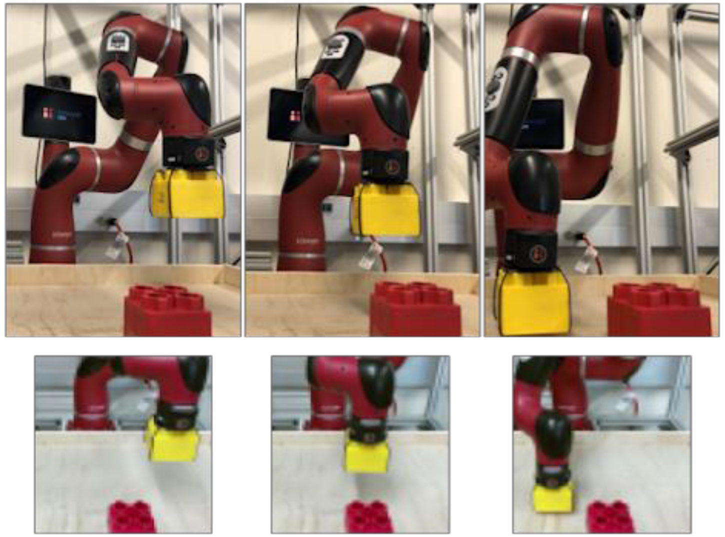
Recent papers:

- Nagabandi et al. Neural Network Dynamics for Model- Based Deep Reinforcement Learning with Model-Free Fine-Tuning.
- Chua et al. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models.
- Feinberg et al. Model-Based Value Expansion for Efficient Model-Free Reinforcement Learning.
- Buckman et al. Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion.

Model-Based RL with Images

What about complex observations?





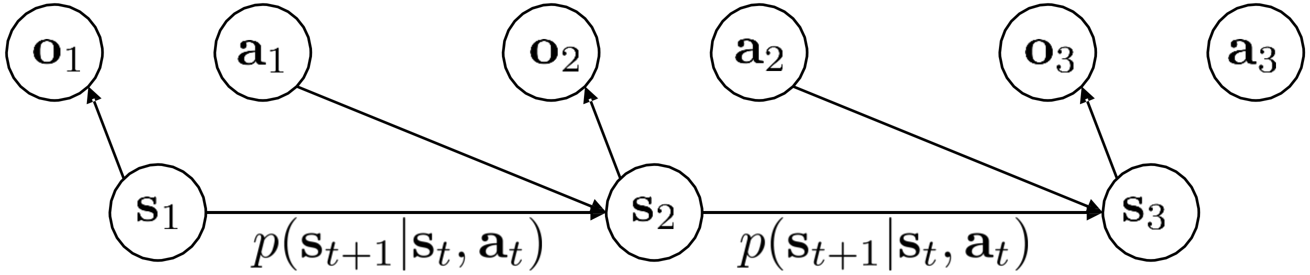
$$f(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{s}_{t+1}$$

What is hard about this?

- High dimensionality
- Redundancy
- Partial observability

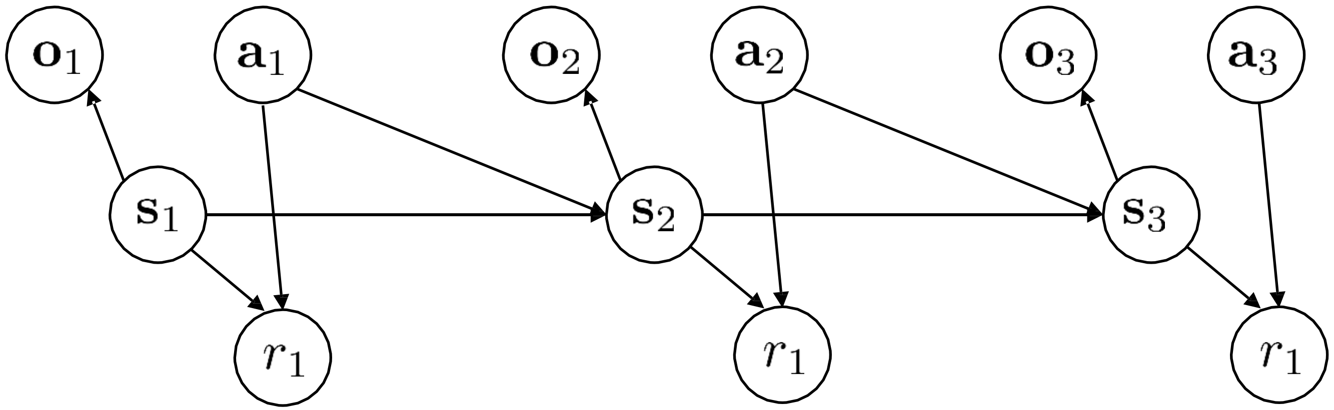
high-dimensional but not dynamic

low-dimension but dynamic



separately learn $p(\mathbf{o}_t|\mathbf{s}_t)$ and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$?

State space (latent space) models



$p(\mathbf{o}_t|\mathbf{s}_t)$ observation model

$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ dynamics model

$p(r_t|\mathbf{s}_t, \mathbf{a}_t)$ reward model

How to train?

standard (fully observed) model: $\max_\phi \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log p_\phi(\mathbf{s}_{t+1,i}|\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$

latent space model: $\max_\phi \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} E\left[\log p_\phi(\mathbf{s}_{t+1,i}|\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \log p_\phi(\mathbf{o}_{t,i}|\mathbf{s}_{t,i})\right]$

expectation w.r.t. $(\mathbf{s}_t, \mathbf{s}_{t+1}) \sim p(\mathbf{s}_t, \mathbf{s}_{t+1}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T})$

Model-based RL with latent space models

$$\max_\phi \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} E\left[\log p_\phi(\mathbf{s}_{t+1,i}|\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \log p_\phi(\mathbf{o}_{t,i}|\mathbf{s}_{t,i})\right]$$

expectation w.r.t. $(\mathbf{s}_t, \mathbf{s}_{t+1}) \sim p(\mathbf{s}_t, \mathbf{s}_{t+1}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T})$

learn *approximate* posterior $q_\psi(\mathbf{s}_t|\mathbf{o}_{1:t}, \mathbf{a}_{1:t})$ "encoder"

many other choices for approximate posterior:

$q_\psi(\mathbf{s}_t, \mathbf{s}_{t+1}|\mathbf{o}_{1:T}, \mathbf{a}_{1:T})$ full smoothing posterior

+ most accurate
- most complicated

$q_\psi(\mathbf{s}_t|\mathbf{o}_t)$ single-step encoder

+ simplest
- least accurate

we'll talk about this one for now

We will discuss variational inference in more detail next week!

Model-based RL with latent space models

$$\max_{\phi} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} E\left[\log p_\phi(\mathbf{s}_{t+1,i}|\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \log p_\phi(\mathbf{o}_{t,i}|\mathbf{s}_{t,i})\right]$$

expectation w.r.t. $\mathbf{s}_t \sim q_\psi(\mathbf{s}_t|\mathbf{o}_t), \mathbf{s}_{t+1} \sim q_\psi(\mathbf{s}_{t+1}|\mathbf{o}_{t+1})$

$q_\psi(\mathbf{s}_t|\mathbf{o}_t)$

simple special case: $q(\mathbf{s}_t|\mathbf{o}_t)$ is *deterministic*

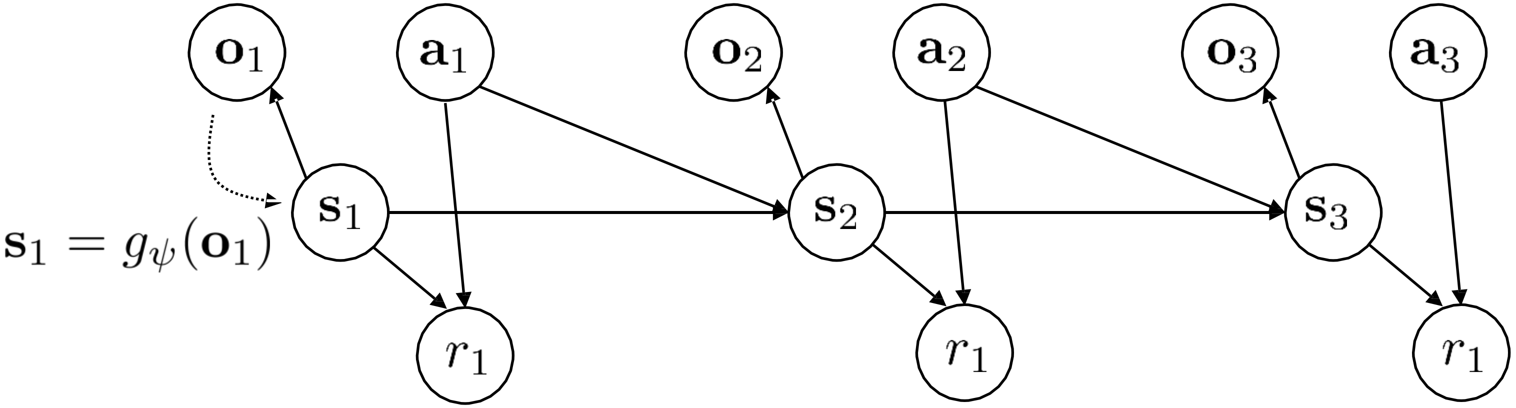
stochastic case requires variational inference (next week)

$q_\psi(\mathbf{s}_t|\mathbf{o}_t) = \delta(\mathbf{s}_t = g_\psi(\mathbf{o}_t)) \Rightarrow \mathbf{s}_t = g_\psi(\mathbf{o}_t)$ deterministic encoder

$$\max_{\phi,\psi} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log p_\phi(g_\psi(\mathbf{o}_{t+1,i})|g_\psi(\mathbf{o}_{t,i}), \mathbf{a}_{t,i}) + \log p_\phi(\mathbf{o}_{t,i}|g_\psi(\mathbf{o}_{t,i}))$$

Everything is differentiable, can train with backprop

Model-based RL with latent space models

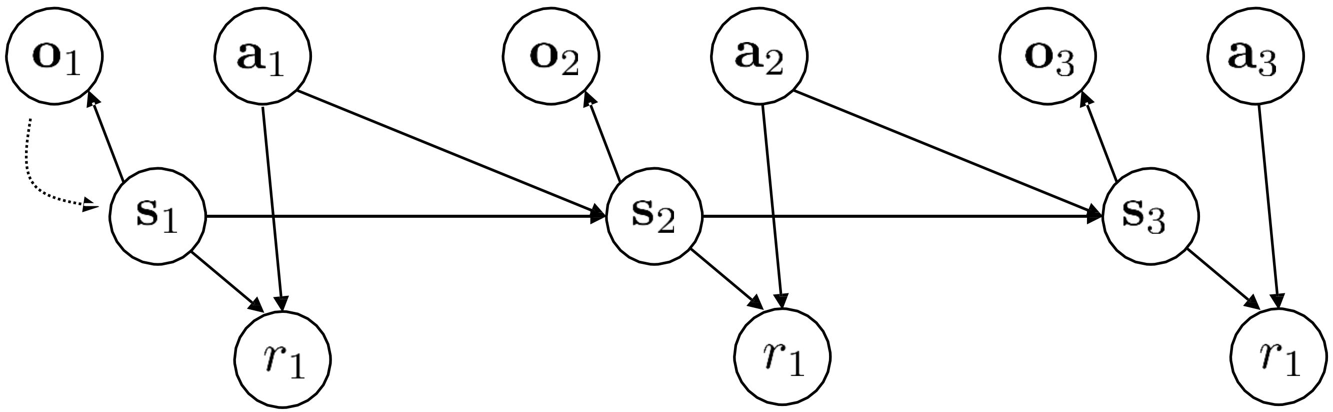


$$\max_{\phi,\psi} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log p_\phi(g_\psi(\mathbf{o}_{t+1,i}) | g_\psi(\mathbf{o}_{t,i}), \mathbf{a}_{t,i}) + \log p_\phi(\mathbf{o}_{t,i} | g_\psi(\mathbf{o}_{t,i})) + \log p_\phi(r_{t,i} | g_\psi(\mathbf{o}_{t,i}))$$

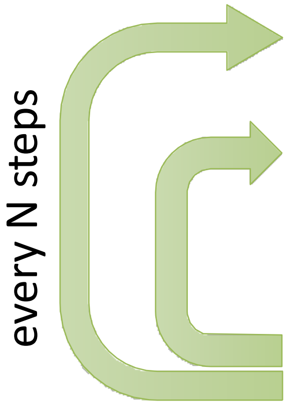
latent space dynamics — image reconstruction — reward model

Many practical methods use a stochastic encoder to model uncertainty

Model-based RL with latent space models



model-based reinforcement learning with latent state:

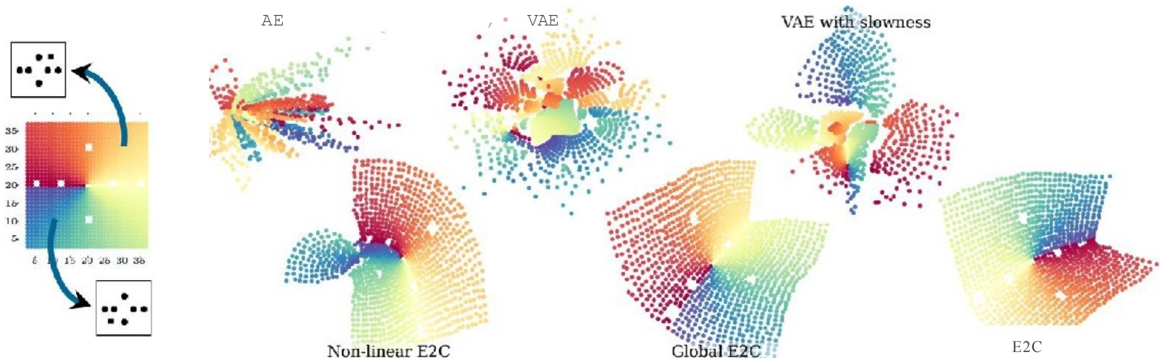


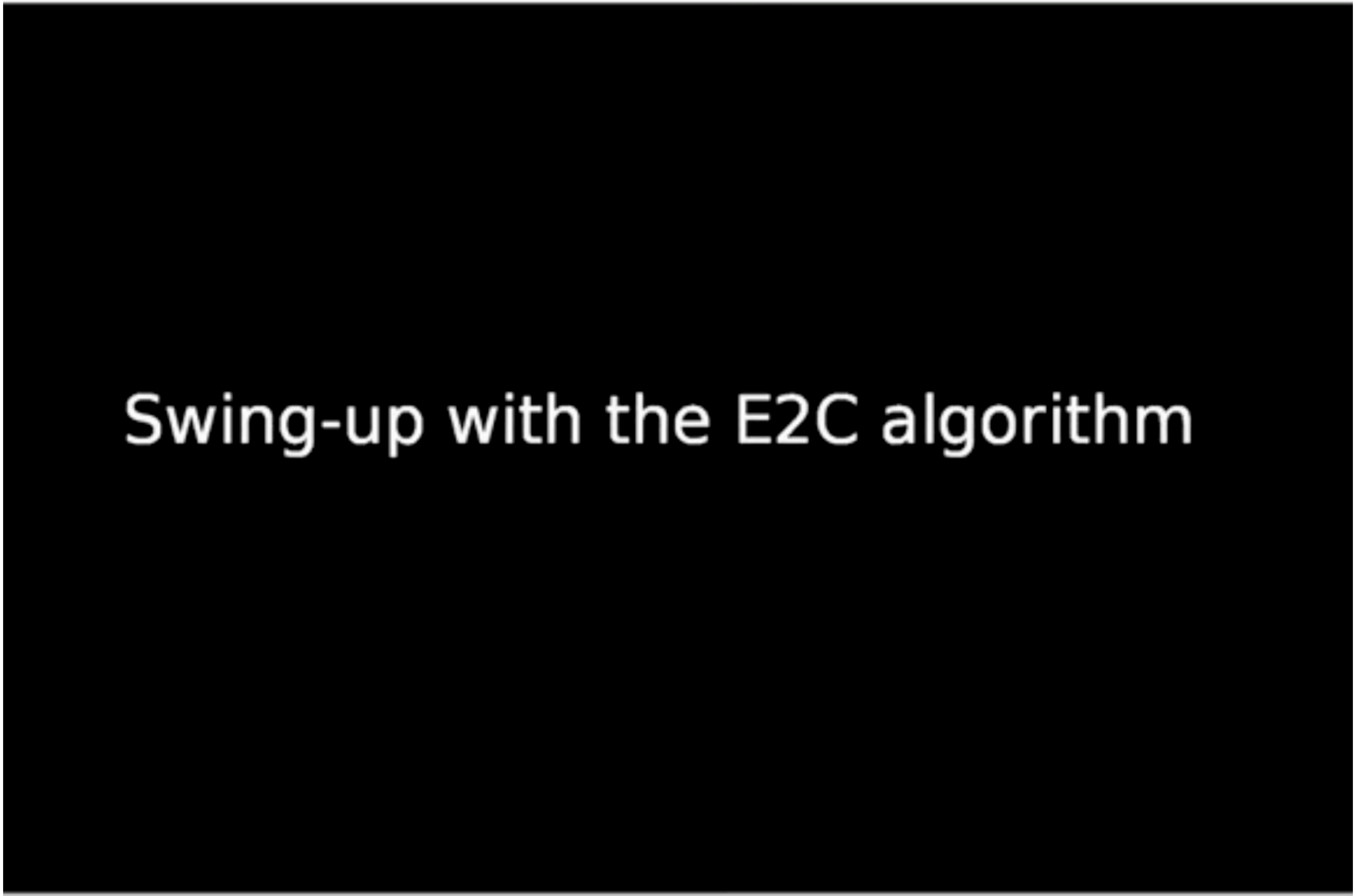
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn $p_\phi(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, $p_\phi(r_t|\mathbf{s}_t)$, $p(\mathbf{o}_t|\mathbf{s}_t)$, $g_\psi(\mathbf{o}_t)$
3. plan through the model to choose actions
4. execute the first planned action, observe resulting $\mathbf{o}'$ (MPC)
5. append $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$ to dataset $\mathcal{D}$

Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images

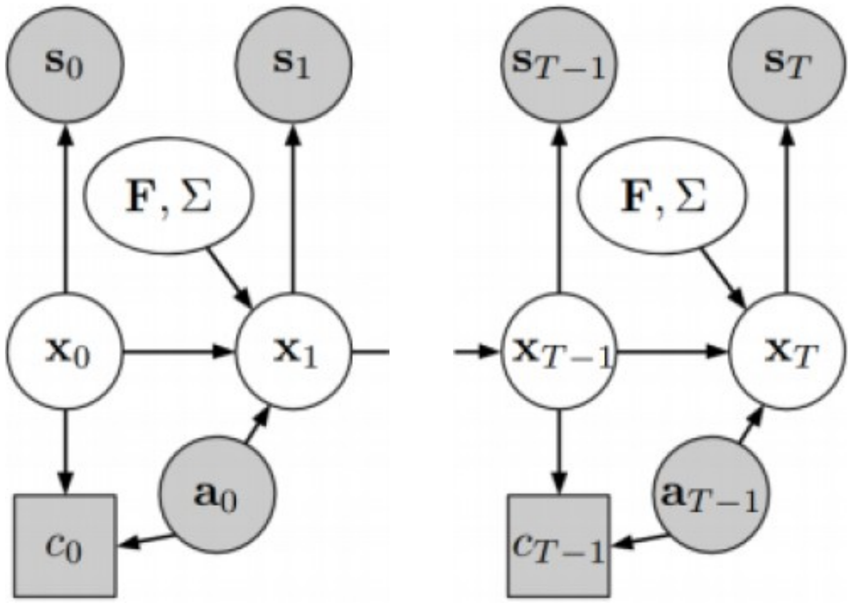
Manuel Watter* Jost Tobias Springenberg*
Joschka Boedecker
University of Freiburg, Germany
{watterm, springj, jboedeck} @cs.uni-freiburg.de

Martin Riedmiller
Google DeepMind
London, UK
riedmiller@google.com




Swing-up with the E2C algorithm

SOLAR: Deep Structured Latent Representations for Model-Based Reinforcement Learning



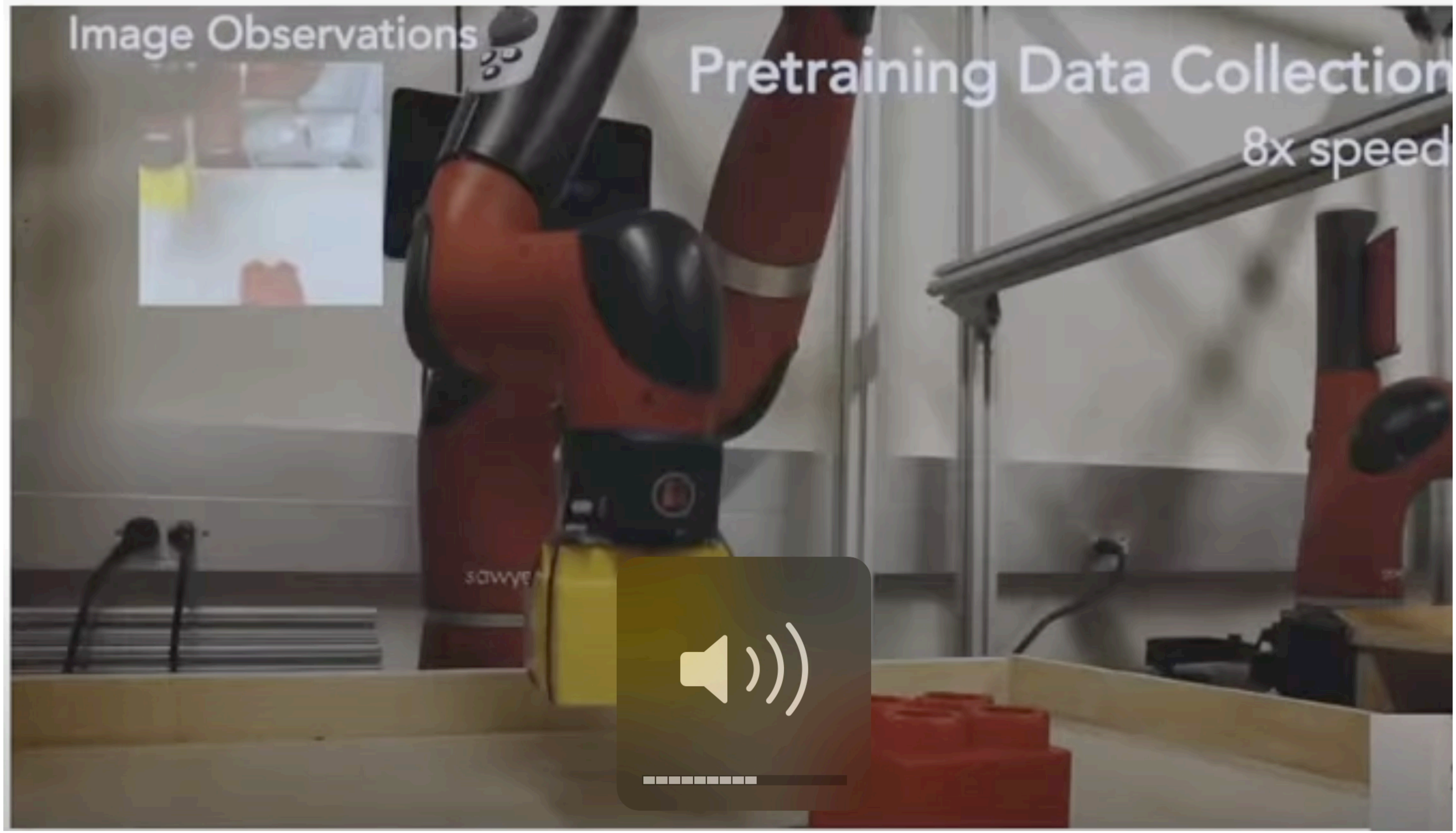
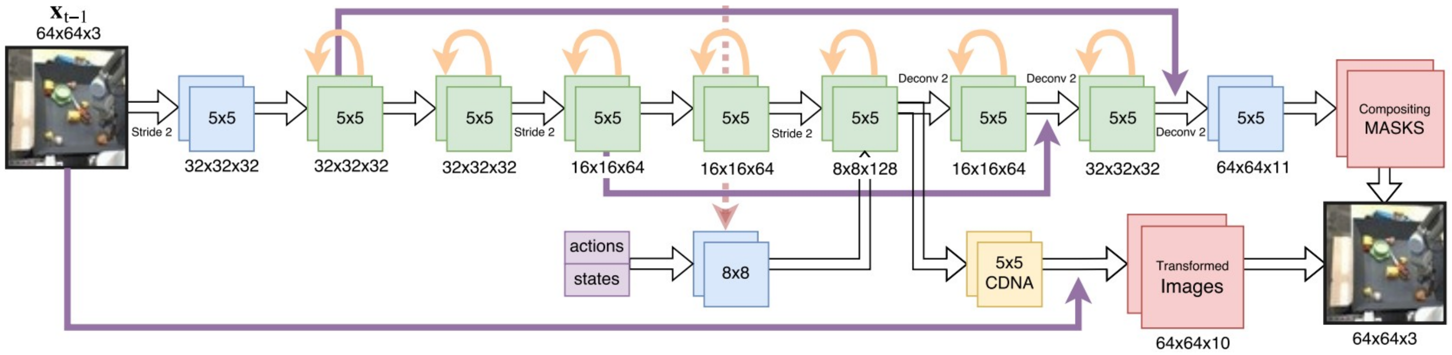


Image Observations
Pretraining Data Collection
8x speed

Learn directly in observation space

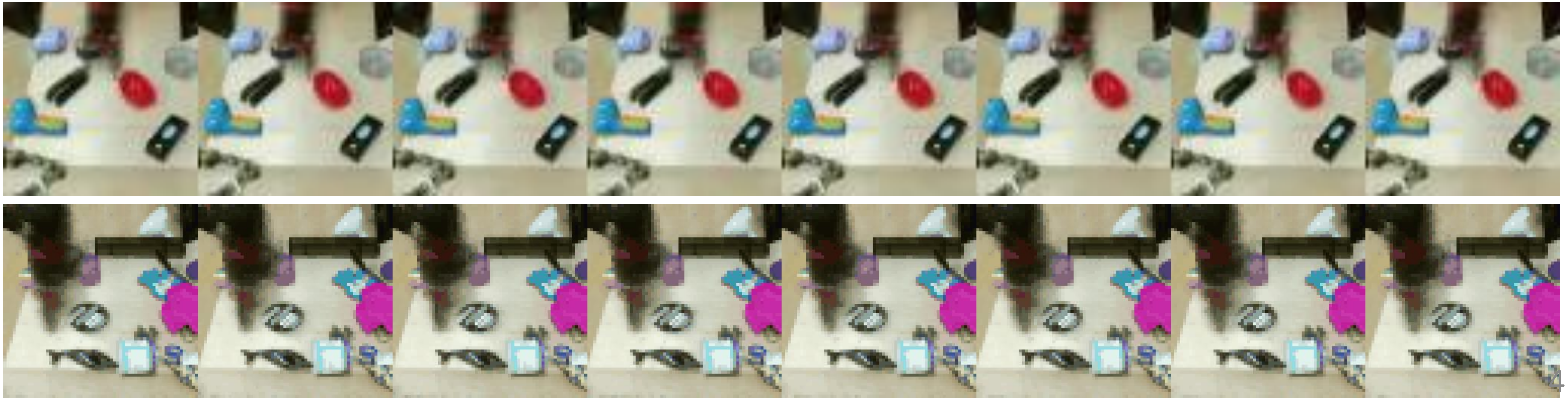
~~**Key idea:** learn embedding $g(\mathbf{o}_t) = \mathbf{s}_t$~~

directly learn $p(\mathbf{o}_{t+1}|\mathbf{o}_t, \mathbf{a}_t)$

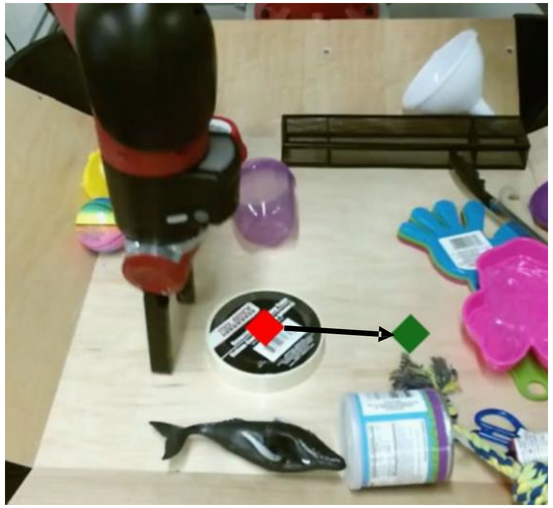


Finn, L. **Deep Visual Foresight for Planning Robot Motion.** ICRA 2017.

Ebert, Finn, Lee, L. **Self-Supervised Visual Planning with Temporal Skip Connections.** CoRL 2017.



Use predictions to complete tasks

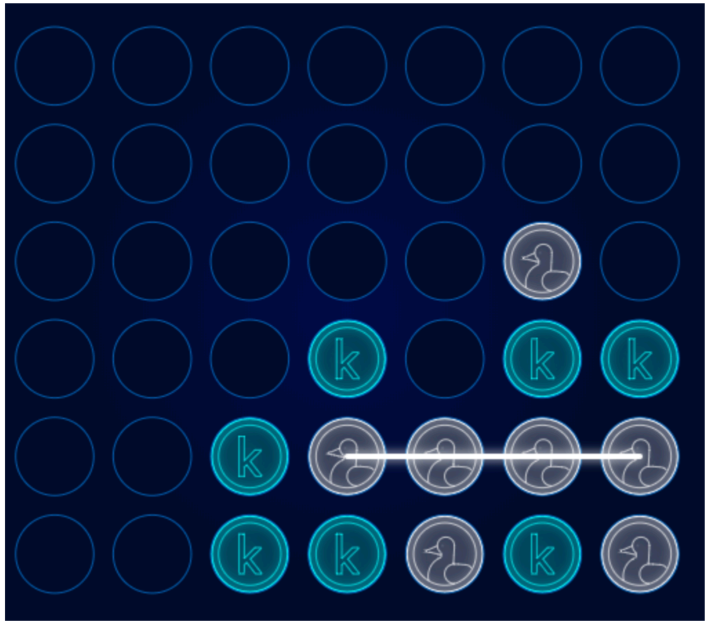


Designated Pixel ◆

Goal Pixel ◆

Task execution

Kaggle competition: Connect X



Due on Sunday 10 March.
Graded on the stabilized version of March 18.

Don't submit after March 10 midnight Paris time, otherwise you will be disqualified.

When you submit, your obtain a score of 600. After you play against 8 people per day, which makes your score evolving several time a day.

Good luck!